INSTITUTO TECNOLÓGICO BELTRÁN
Centro de Tecnología e Innovación

# Algoritmo FOIL

**Institución:** Instituto Tecnológico Beltrán

**Materia:** Aprendizaje Automático

**Docente:** Yanina Scudero

**Integrantes:** Marco Medina

**Fecha:** 17/09

## Actividad 1:

**Nuestro DataSet:**

```
datos = [
{"edad": 22, "departamento": "IT", "nivel_educativo": "terciario", "en_formacion": True},
{"edad": 24, "departamento": "IT", "nivel_educativo": "universitario", "en_formacion": True},
{"edad": 21, "departamento": "RRHH", "nivel_educativo": "terciario", "en_formacion": True},
{"edad": 35, "departamento": "IT", "nivel_educativo": "universitario", "en_formacion": False},
{"edad": 40, "departamento": "Finanzas", "nivel_educativo": "maestría", "en_formacion": False},
{"edad": 29, "departamento": "RRHH", "nivel_educativo": "universitario", "en_formacion": False},
{"edad": 23, "departamento": "IT", "nivel_educativo": "terciario", "en_formacion": True},
{"edad": 38, "departamento": "Finanzas", "nivel_educativo": "universitario", "en_formacion": False}]
```

1. Separar los ejemplos positivos (en_formacion == True) y negativos (en_formacion == False).

    - Ya teniendo nuestro código el cual utiliza la forma más básica del algoritmo de FOIL define las reglas que se utilizaran para reconocer los ejemplos positivos

```
positivos = [p for p in datos if p["en_formacion"]]
negativos = [p for p in datos if not p["en_formacion"]]



def inducir_regla(positivos, negativos):
    atributos = ["edad", "departamento", "nivel_educativo"]
    regla = {}

    for atributo in atributos:
        valores_pos = set(p[atributo] for p in positivos)
        valores_neg = set(p[atributo] for p in negativos)

        if atributo == "edad":
            valores_validos = [v for v in valores_pos if v not in valores_neg]
        else:
            valores_validos = list(valores_pos - valores_neg)


        if atributo == "departamento":
            regla[atributo] = list(valores_pos)
        elif valores_validos:
            regla[atributo] = valores_validos

    return regla
```

- Podemos visualizar que al principio decidimos separar los diccionarios en los cuales se respeta que el empleado tenga el atributo “en_formacion” = True
- Luego en la función recibe la lista de diccionarios con el atributo “en_formacion” = True Y “en_formacion” = False. Esto lo realiza para encontrar en ella los atributos únicos.

Utilizando esta Lógica se pudo identificar los siguientes atributos únicos que forman a los empleados que se encuentran en formación (“en_formacion” = True).

```
regla inducida para identificar a un alumno:
edad debe ser uno de: [24, 21, 22, 23]
departamento debe ser uno de: ['RRHH', 'IT']
nivel_educativo debe ser uno de: ['terciario']
```

**En conclusión:**

Al utilizar la idea más básica del algoritmo FOIL pude identificar las reglas que comprenden los casos válidos en nuestro set de datos. El algoritmo util para demostrar atributos unicos. Por ejemplo: el rango de edad en los empleados (21-24 años) y el nivel educativo terciario son los mejores predictores para identificar personas en formación, también se pudo comprobar que en el caso del atributo “departamento” no es realmente discriminatorio para los empleados en formación

## Actividad 2:

Desarrollar un programa en Python que obtenga el cálculo del FOIL Gain para la condición nivel_educativo == 'terciario', con la siguiente salida:

```
Condición: nivel_educativo == 'terciario'
P (positivos antes) = 4
N (negativos antes) = 4
p (positivos después) = β
n (negativos después) = 0
p / (p + n) = 1.000
P / (P + N) = 0.500
log2(p / (p + n)) = 0.000
log2(P / (P + N)) = -1.000
FOIL Gain = 3.000
```

- Para lograr llegar a esta salida, hicimos una funcion en la cual se toma en cuenta el “total de positivos” y “total de negativos”, antes y despues de aplicar una condición. Esto para tener los datos necesario para realizar la Formula de Foil Gain y saber qué tan buena es una condición para clasificar ejemplos.

    - Fórmula:

$$\text{FOIL Gain} = p \times \left( \log_2 \left( \frac{p}{p+n} \right) - \log_2 \left( \frac{P}{P+N} \right) \right)$$

Codigo:

```python
import math

def calcular_foil_gain(datos, condicion_atributo, condicion_valor):
    # Contadores iniciales (antes de aplicar la condición)
    P = sum(1 for p in datos if p["en_formacion"])
    N = sum(1 for p in datos if not p["en_formacion"])

    # Contadores después de aplicar la condición
    p = sum(1 for p in datos if p[condicion_atributo] == condicion_valor and p["en_formacion"])
    n = sum(1 for p in datos if p[condicion_atributo] == condicion_valor and not p["en_formacion"])

    # Cálculos para FOIL Gain
    def log2_safe(x):
      return math.log2(x) if x > 0 else float('-inf')

    foil_gain = p * (log2_safe(p / (p + n)) - log2_safe(P / (P + N)))

    # Mostrar resultados
    print(f"P = {P}, N = {N}")
    print(f"p = {p}, n = {n}")
    print(f"p / (p + n) = {p / (p + n):.3f}")
    print(f"P / (P + N) = {P / (P + N):.3f}")
    print(f"log2(p / (p + n)) = {log2_safe(p / (p + n)):.3f}")
    print(f"log2(P / (P + N)) = {log2_safe(P / (P + N)):.3f}")
    print(f"FOIL Gain = {foil_gain:.3f}")

    return foil_gain

# Calcular FOIL Gain para nivel_educativo == 'terciario'
foil_gain = calcular_foil_gain(datos, "nivel_educativo", "terciario")
```

- Es importante recalcar que utilizamos una librería que nos permite hacer cálculos matemáticos con los contadores que se utilizaron antes y después de aplicar una condición. Todo esto para que nos de la siguiente salida y coincida con la solicitada en la actividad.

```
P = 4, N = 4
p = 3, n = 0
p / (p + n) = 1.000
P / (P + N) = 0.500
log2(p / (p + n)) = 0.000
log2(P / (P + N)) = -1.000
FOIL Gain = 3.000
```

## **ACTIVIDAD 3:** Utiliza la condición edad <= 23

En este caso decidimos realizar unos pequeños cambios en el código para utilizar la fórmula de FOIL Gain bajo la condición de eda <= 23.
Esto lo hicimos cambiando los parametros de la función y especificar la condición en el cuerpo del código.

```python
import math

def calcular_edad_foil_gain(datos, edad_limite):
    # Contadores iniciales (antes de aplicar la condición)
    P = sum(1 for p in datos if p["en_formacion"])
    N = sum(1 for p in datos if not p["en_formacion"])

    # Contadores después de aplicar la condición
    p = sum(1 for p in datos if p["edad"] <= edad_limite and p["en_formacion"])
    n = sum(1 for p in datos if p["edad"] <= edad_limite and not p["en_formacion"])

    # Cálculos para FOIL Gain
    def log2_safe(x):
      return math.log2(x) if x > 0 else float('-inf')

    foil_gain = p * (log2_safe(p / (p + n)) - log2_safe(P / (P + N)))

    # Mostrar resultados
    print(f"La condicion: edad <= {edad_limite}")
    print(f"P = {P}, N = {N}")
    print(f"p = {p}, n = {n}")
    print(f"p / (p + n) = {p / (p + n):.3f}")
    print(f"P / (P + N) = {P / (P + N):.3f}")
    print(f"log2(p / (p + n)) = {log2_safe(p / (p + n)):.3f}")
    print(f"log2(P / (P + N)) = {log2_safe(P / (P + N)):.3f}")
    print(f"FOIL Gain = {foil_gain:.3f}")

    return foil_gain

    # Calcular FOIL Gain para "edad" =< 23
foil_gain = calcular_edad_foil_gain(datos, 23)
```

Lo que nos retorna los siguientes datos.

```
La condicion: edad <= 23
P = 4, N = 4
p = 3, n = 0
p / (p + n) = 1.000
P / (P + N) = 0.500
log2(p / (p + n)) = 0.000
log2(P / (P + N)) = -1.000
FOIL Gain = 3.000
```